*Département Informatique : Deuxième Année*

---

**Décidabilité T.D. N° 1**

29 décembre 2010

**Elements de base** [1]

# 1 Algorithmes et complexité

Dans cet exercice, nous considérons un graphe G = (V,E) et nous étudions les deux problèmes classiques suivants :

– Accessibilité : étant donnés deux sommets s1 et s2 de G, existe-t-il un chemin allant de s1 à s2.
– Recherche d'un circuit dans G : étant donnés deux sommets s1 et s2 de G, existe-t-il un circuit passant par s1 et s2 ?

1. Trouver un algorithme pour résoudre le premier problème.

   **Corrigé** Une méthode pour la résolution du problème consiste à partir du sommet s1, parcourir le graphe jusqu'à atteindre le sommet s2 ou fin de parcours (le sommet s2 n'a pas été trouvé). Il existe plusieurs façons de parcourir un graphe (largeur d'abord, profondeur d'abord,etc.).
2. Calculer la complexité de cet algorithme.

   **Corrigé** Au pire l'algorithme traverse tous les arcs du graphe avant de conclure, et au pire il y a $n^2$ arcs (où n est le nombre de sommets). La complexité est donc de $O(n^2)$.
3. En déduire un algorithme pour la résolution du deuxième problème et calculer sa complexité.

   **Corrigé** Le deuxième problème peut être traduit comme suit : il existe un circuit contenant s1 et s2 si s2 est accessible à partir de s1 et que s1 est accessible à partir de s2. La complexité est donc de $O(n^2)$.

---

1. Ce TD est une adaptation d'un texte qui a été proposé par Houcine Senoussi.

# 2 A la découverte du problème SAT

Dans cet exercice nous nous intercessions aux formules bien formées mises sous forme conjonctive normale (fcn), dans lesquelles chaque clause a exactement deux littéraux. Autrement dit, nos formules sont de la forme : $F = C_1 \wedge C_2 ... \wedge C_n$ avec $Ci = (x_{i1} \vee x_{i2})$ , $x_{i1}$ et $x_{i2}$ étant deux littéraux. Pour étudier une telle formule, nous lui associons un graphe orienté G(F) défini comme suit :

- Les sommets du graphe sont les littéraux.
- Les arêtes sont définies comme suit :
  - Pour chaque clause de la forme $(\neg p \vee q)$,le graphe contient les arêtes $(p, q)$ et $(\neg q, \neg p)$.
  - Pour chaque clause de la forme $(p \vee q)$,le graphe contient les arêtes $(\neg p, q)$ et $(\neg q, p)$.
  - Pour chaque clause de la forme $(\neg p \vee \neg q)$,le graphe contient les arêtes $(p, \neg q)$ et $(q, \neg p)$.

1. Prenons $F = (p \vee q) \wedge (q \vee r) \wedge (\neg q \vee s) \wedge (\neg p \vee t) \wedge (\neg t \vee \neg s) \wedge (\neg r \vee s)$.

   (a) Etudier la satisfiabilité de F en utilisant la table de vérité.

   (b) Représenter le graphe G(F).

   **Corrigé**
   La table de vérité et le graphe doivent permettre de constater que la formule est satisfiable ssi Le graphe ne possède pas de cycle passant par une proposition et sa négation.

2. Démontrez la propriété suivante : Soit M un modèle de F, pour toute arête $(l_1, l_2)$de G(F), nous avons : Si dans M $l_1$ vaut VRAI alors $l_2$ vaut VRAI aussi.

   **Corrigé** Un arc (l1,l2) dans le graphe signifie qu'il y a une clause $\neg l_1 \vee l_2$ dans la formule. Donc : dans un modèle, la formule vaut VRAI, chaque clause vaut VRAI, puisque $l_1$ vaut VRAI alors $l_2$ vaut obligatoirement VRAI.

3. Généralisiez la propriété précédente au cas de deux littéraux $l_1$ et $l_2$ tels qu'il existe un chemin allant de $l_1$ à $l_2$.

   **Corrigé** Un chemin (l1 ,l2 ) signifie une suite d'arcs $(l_1, s_2), (s_2, s_3), ..., (s_{n?1}, s_n), (s_n, l_2)$. Donc de proche en proche et d ?après la question précédente : dans un modèle si $l_1$ vaut VRAI, $s_1$ vaut VRAI, $s_2$ vaut VRAI,etc. et enfin $l_2$ vaut VRAI.

4. Déduisez une condition nécessaire et suffisante sur G(F) pour que F soit unsatisfiable.

   **Corrigé** F est unsatisfiable ssi il existe un cycle passant par une proposition p et sa négation $\neg p$.

5. Conclure de ce qui précède qu'il existe un algorithme de résolution de ce problème et donner sa complexité.

**Corrigé** Nous avons transformé le problème 2-SAT en un problème de recherche de cycle dans un graphe. Ce dernier problème a une complexité polynomiale d ?après l ?exercice précédent. Il en est donc de même pour le problème 2-SAT.

Le problème que nous venons d'étudier est le problème 2-SAT. Ce problème est un cas particulier d'un problème plus général : le problème SAT dans lequel le nombre de littéraux dans chaque clause est quelconque.

La méthode de résolution du problème 2-SAT est-elle généralisable au problème SAT ?

**Corrigé** Cette méthode n ?est pas généralisable aux problèmes n-SAT puisque le construction du graphe suppose des clauses à deux littéraux. Nous verrons plus tard que n-SAT n ?est pas un problème polynomial.

# 3 Machine de Turing

Soit la machine de Turing définie ainsi :
- $Q = \{q_0, q_1, q_2, q_e, q_s\}$
- $\Sigma = \Gamma \setminus \{\diamond\} = \{0, 1\}$
- $F = \{q_s\}$
- $\delta(q_0, 0) = (q_0, 0, R)$, $\delta(q_0, 1) = (q_0, 1, R)$, $\delta(q_0, \diamond) = (q_0, \diamond, L)$,
- $\delta(q_1, 0) = (q_2, \diamond, L)$, $\delta(q_0, 1) = (q_e, \diamond, L)$,
- $\delta(q_2, 0) = (q_s, \diamond, L)$, $\delta(q_2, 1) = (q_e, \diamond, L)$, $\delta(q_2, 1) = (q_e, \diamond, L)$.

Faire tourner cette machine de Turing sur les exemples suivants :
- 111011001
- 111011000
- 111011010

Trouver le problème que cette machine de Turing permet de résoudre.

**Corrigé** le problème des chaines se terminant par 00.

# 4 Machine de Turing & Langage

Nous avons vu que le langage formé des mots $a^n b^n$ avec a et b deux symboles et n un entier est un langage hors-contexte donc dont les mots peuvent être reconnus par un automate à pile. Trouver une machine de Turing permettant de reconnaître les mots de ce langage.

**Corrigé**
- $Q = \{q_1, q_2, q_3, q_4, q_5, q_s, q_e\}$
- $\Gamma = \{a, b, X, Y, \diamond, \}$
- $\Sigma = \{a, b\}$
- $F = \{q_s\}$

– $\delta$ contient :

1. $(q_1, a) \rightarrow (q_2, X, R)$, $(q_1, b) \rightarrow (q_e, b, R)$
   $(q_1, X) \rightarrow (q_e, X, R)$, $(q_1, Y) \rightarrow (q_4, Y, R)$, $(q_1, \diamond) \rightarrow (q_s, \diamond, R)$.

2. $(q_2, a) \rightarrow (q_2, a, R)$, $(q_2, b) \rightarrow (q_3, Y, L)$
   $(q_2, X) \rightarrow (q_e, X, R)$, $(q_2, Y) \rightarrow (q_2, Y, R)$, $(q_2, \diamond) \rightarrow (q_e, \diamond, R)$.

3. $(q_3, a) \rightarrow (q_3, a, L)$, $(q_3, b) \rightarrow (q_e, b, L)$
   $(q_3, X) \rightarrow (q_1, X, R)$, $(q_3, Y) \rightarrow (q_3, Y, L)$, $(q_3, \diamond) \rightarrow (q_e, \diamond, R)$.

4. $(q_4, a) \rightarrow (q_e, a, L)$, $(q_4, b) \rightarrow (q_e, b, L)$
   $(q_4, X) \rightarrow (q_e, X, R)$, $(q_4, Y) \rightarrow (q_4, Y, R)$, $(q_4, \diamond) \rightarrow (q_s, \diamond, R)$.